Univariate Volatilty

Kevin Sheppard

Today

- Realized Variance
- Bid-Ask Bounce
- Volatility Signature Plots
- Alternative Estimators
- Modeling and Forecasting RV
- Implied Variance

Realized Variance

- Use UHF data to precisely estimate variance over a given period
  - Typically 1 day
  - Avoids diurnal effects

$$RV_t^{(m)} = \sum_{i=1}^{m} (p_{i,t} - p_{i-1,t})^2 = \sum_{i=1}^{m} r_{i,t}^2$$

- Like variance estimator except no need to demean
- Assumption is that $m \to \infty$
- Estimated *integrated variance*

$$\lim_{m \to \infty} RV_t^{(m)} = \int_t^{t+1} \sigma_s^2 ds$$

Simulated Brownian Motion

- Assume μ and σ are annualized

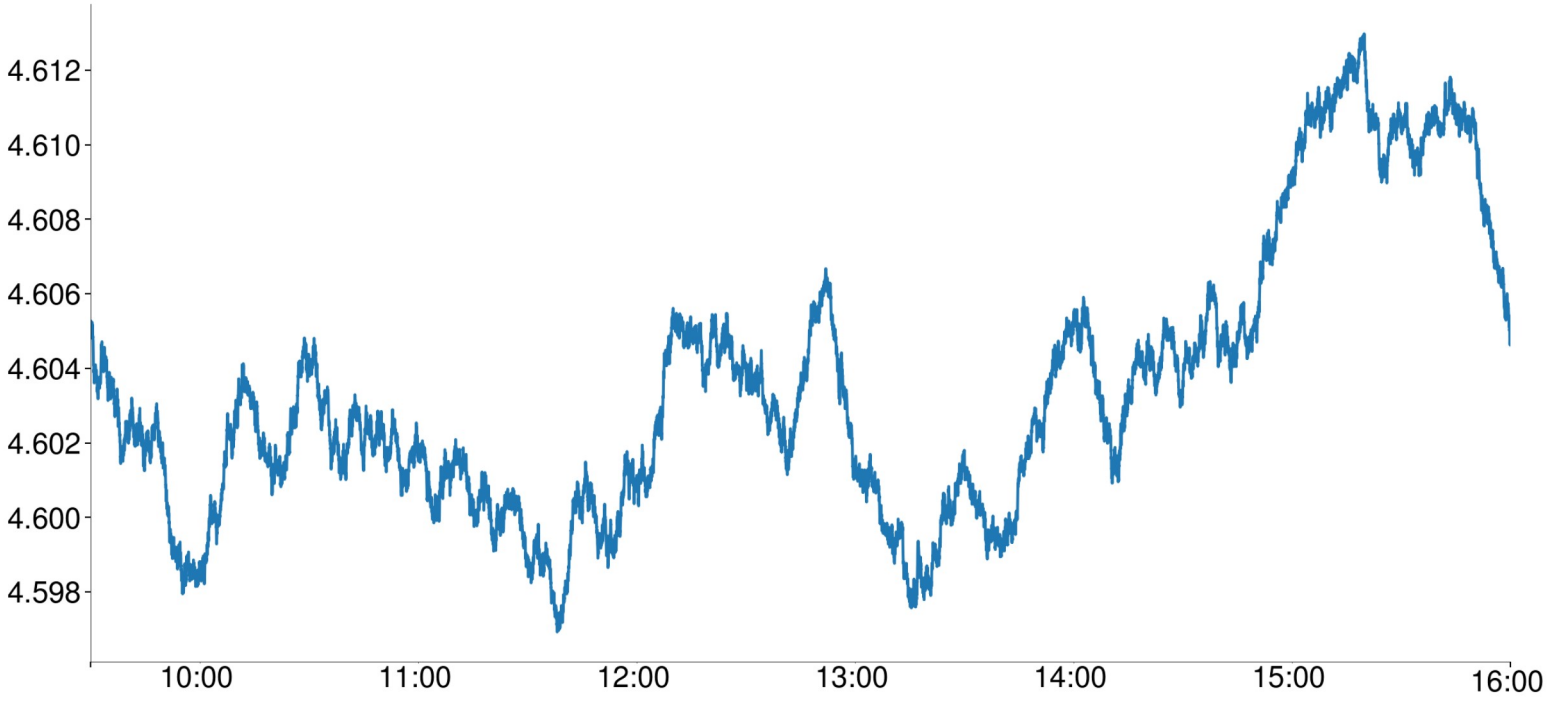
$$r_{i,t} \overset{iid}{\sim} N\left(\frac{\mu}{252m}, \frac{\sigma^2}{252m}\right)$$

$$p_{i,t} = p_{i-1,t} + r_{i,t}$$

- $p_{0,t} = \ln(100)$

The price path

In [3]:
```
plot(p)
```



Implementing RV

In [4]:

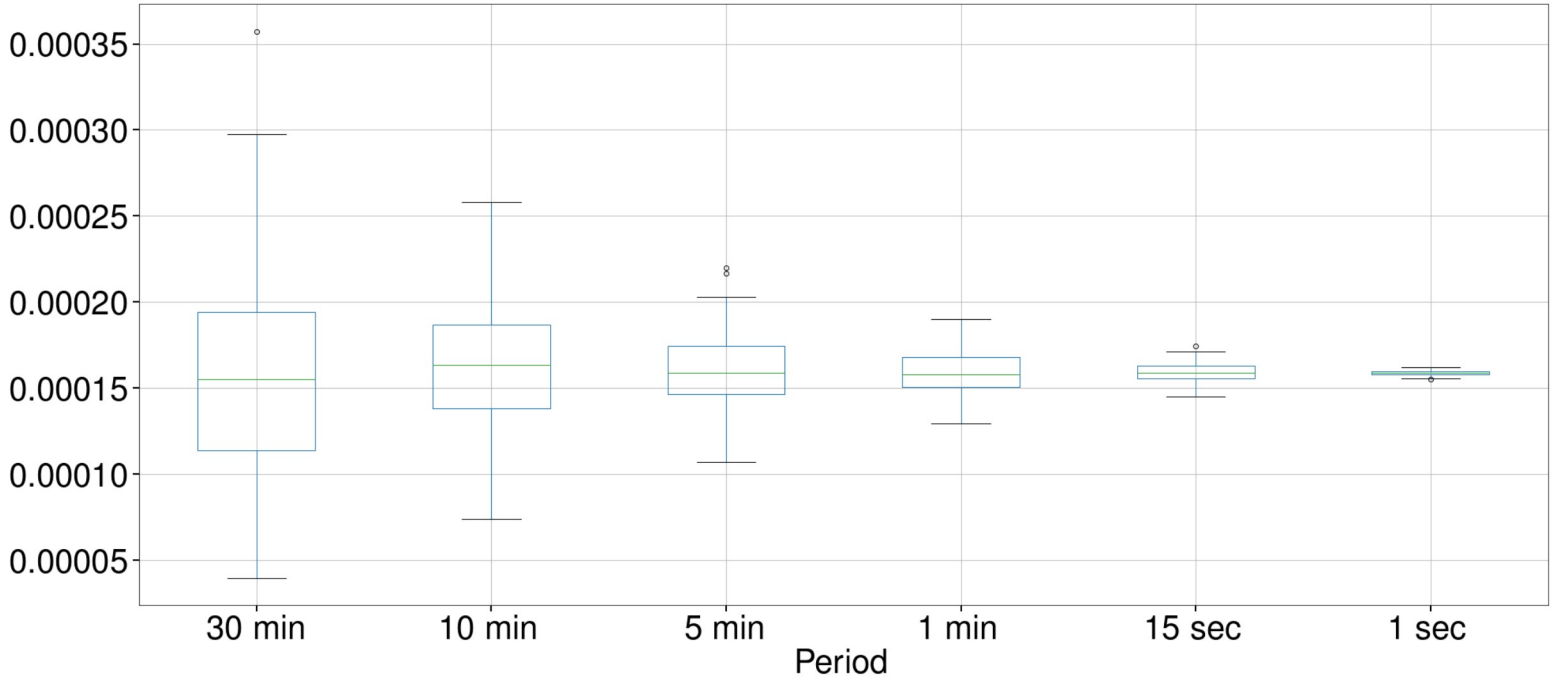
```
rv = {}

for i in (13,39,78,390,1560,23400):
    step = m // i
    rets = p.iloc[::step].diff().dropna()
    _rv = (rets**2).sum()
    rv[step] = _rv
np.sqrt(252 * pd.Series(rv))
```

Out[4]:

```
1800    0.210890
600     0.242003
300     0.209546
60      0.180562
15      0.197455
1       0.200257
dtype: float64
```

RV estimation error

In [6]:
```
plot_rv()
```



Bid-Ask Bounce

- Primary challenge of RV implementation
- Not one price, but two
    - Bid
    - Ask or Offer
- Transactions tend to bound between these two prices
- Creates many high-frequency noise returns
- Observed return is

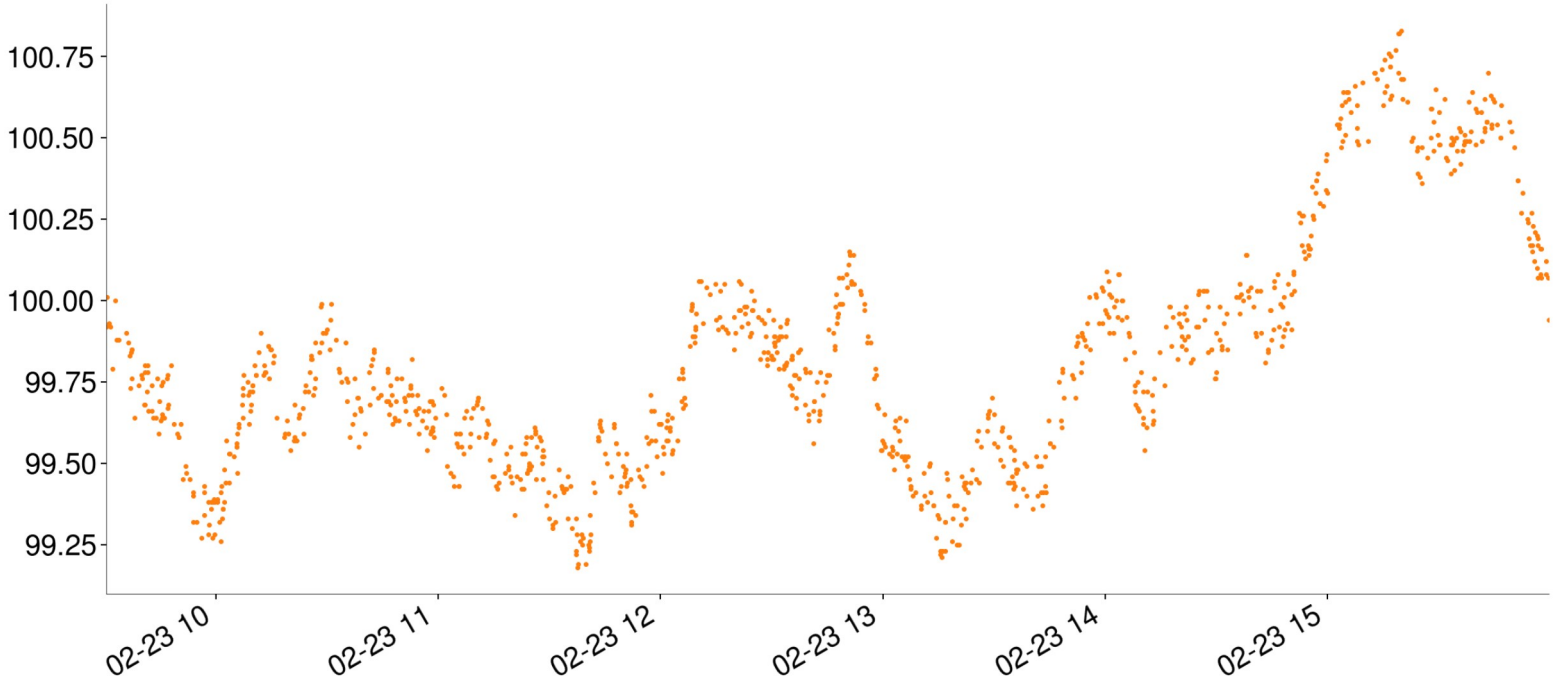
$$r_{i,t} = r^{\star}_{i,t} + \epsilon_{i,t}$$

- $r^{\star}_{i,t}$ is the return we would like see
- $\epsilon_{i,t}$ follows an MA(1)

$$RV_t^{(m)} \approx \widehat{RV}_t + m\tau^2$$

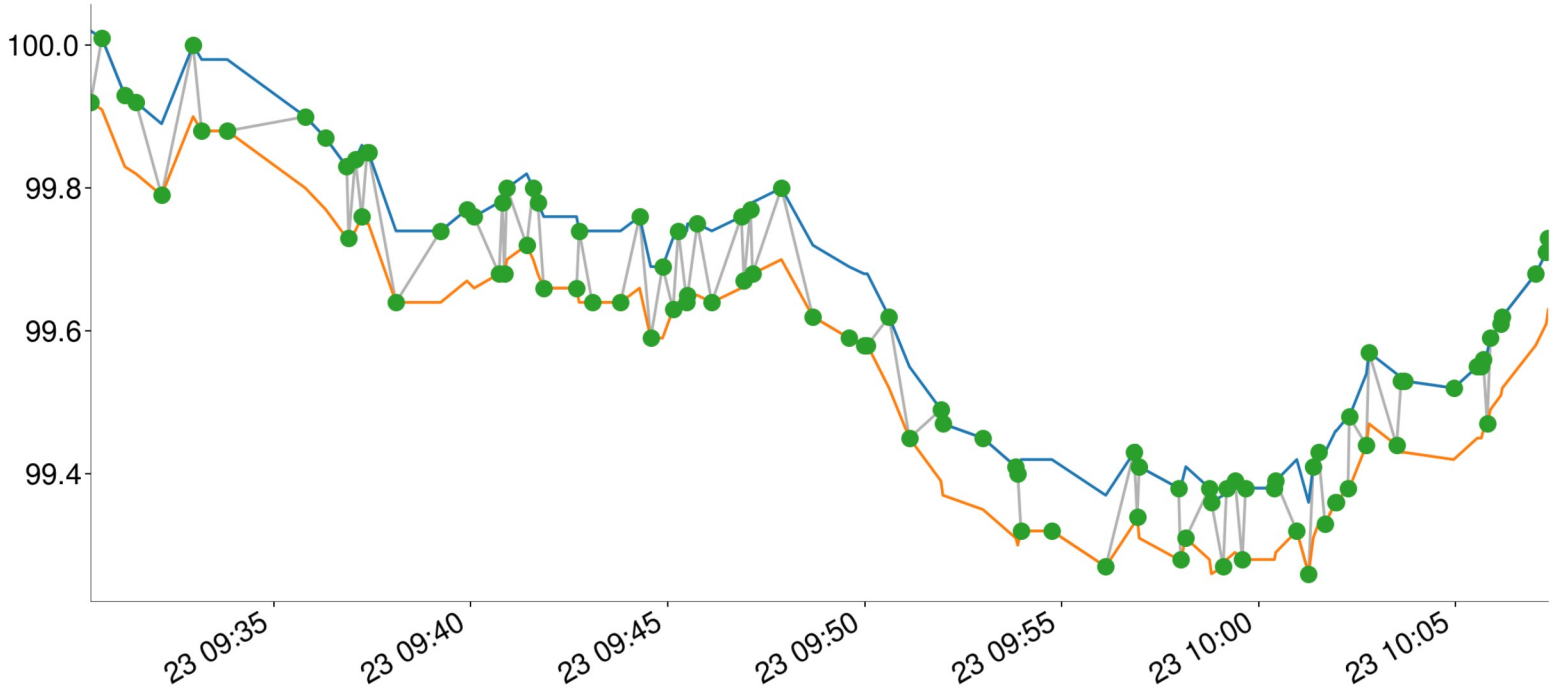
Realistic UHF Asset Price

In [8]:
```
plot_trans()
```



Transaction, Bid, and Ask Prices

In [9]:
```
plot_trans(zoom=True)
```



Volatilty Signature Plots

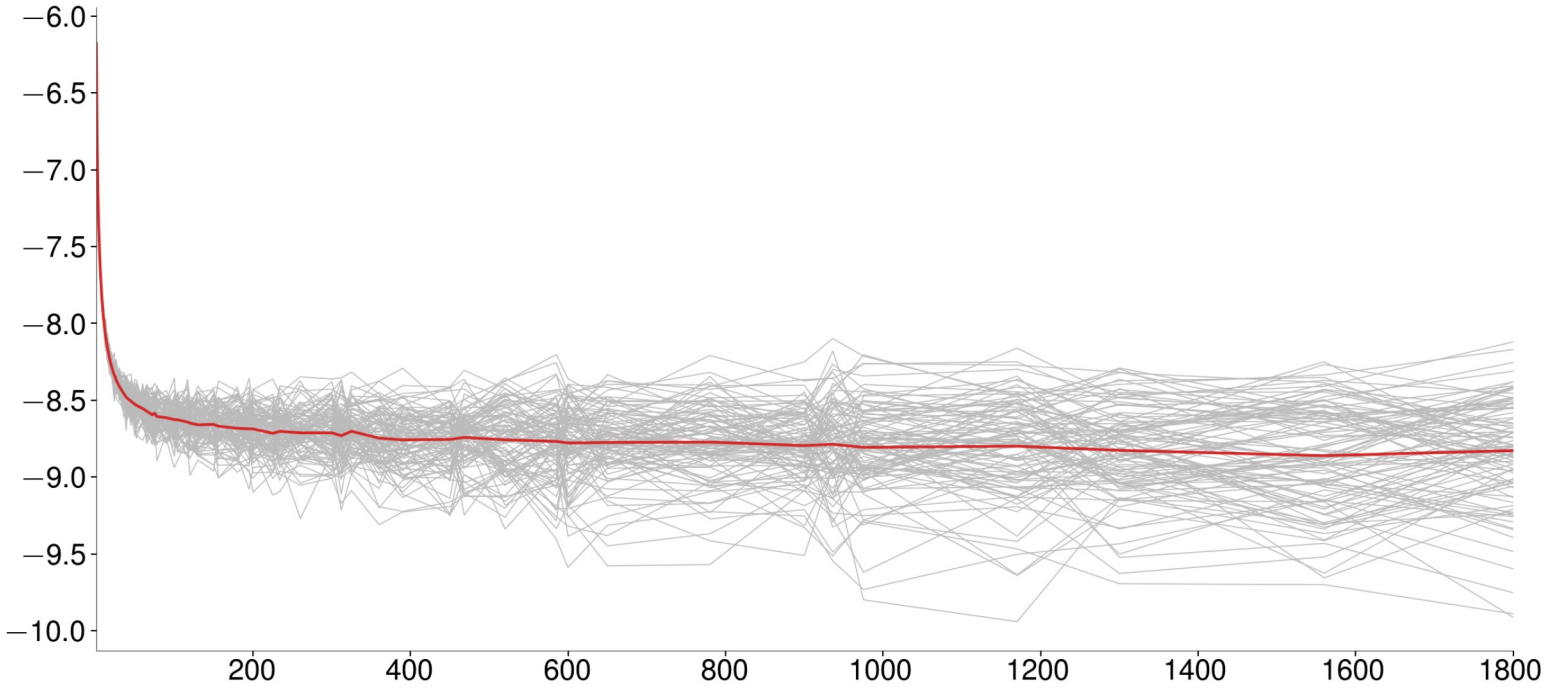
- Time-series average of Realized Variance

$$\overline{RV}_t^{(m)} = T^{-1} \sum_{t=1}^{T} RV_t^{(m)}$$

- Function of the number of samples m
- Look for inflection points

Simulated Data Signature Plot

In [11]:
```
vsp()
```



Sparse sampling and Subsampling

- Simple solution is to use sparse sampling
- Problem: there are many 5-minute returns between 9:30 and 16:00
- Solution: Use them all (or many since *all* is overkill)
    - 9:30-9:35
    - 9:31:9:36
    - 9:32-9:37
    - ...
- Mild improvement in accuracy

Robustifying Realized Variance

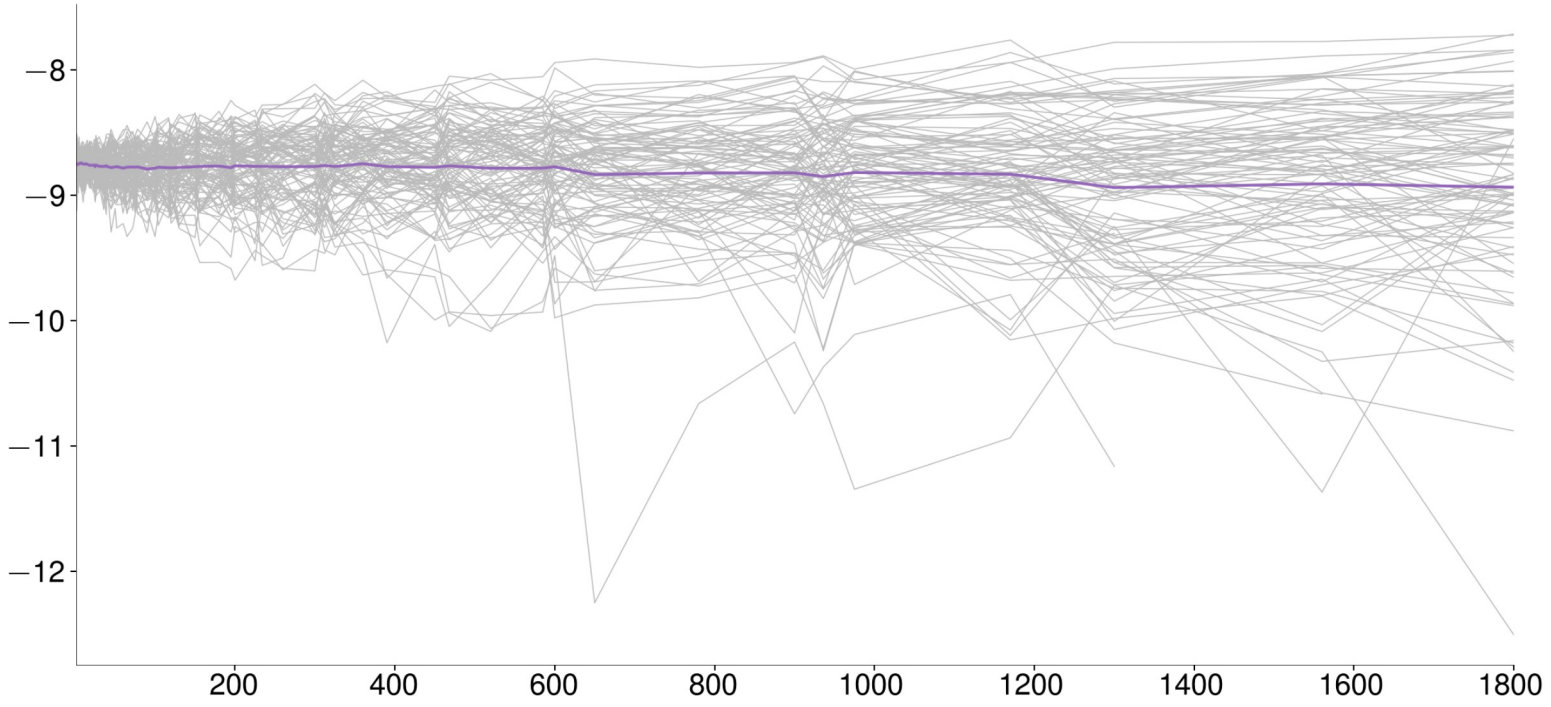
- Bid-ask bounce produces bias
- Best solution is to use an estimator that is robust to noise

$$RV_t^{AC1(m)} = \sum_{i=1}^{m} r_{i,t}^2 + 2\sum_{i=2}^{m} r_{i,t} r_{i-1,t}$$

- Removes first order bias
- Not consistent, need more lags for consistency
- *Realized Kernel* is like the Newey-West estimator

AC1 Volatility Signature Plot

In [13]:
```
vsp_ac1()
```

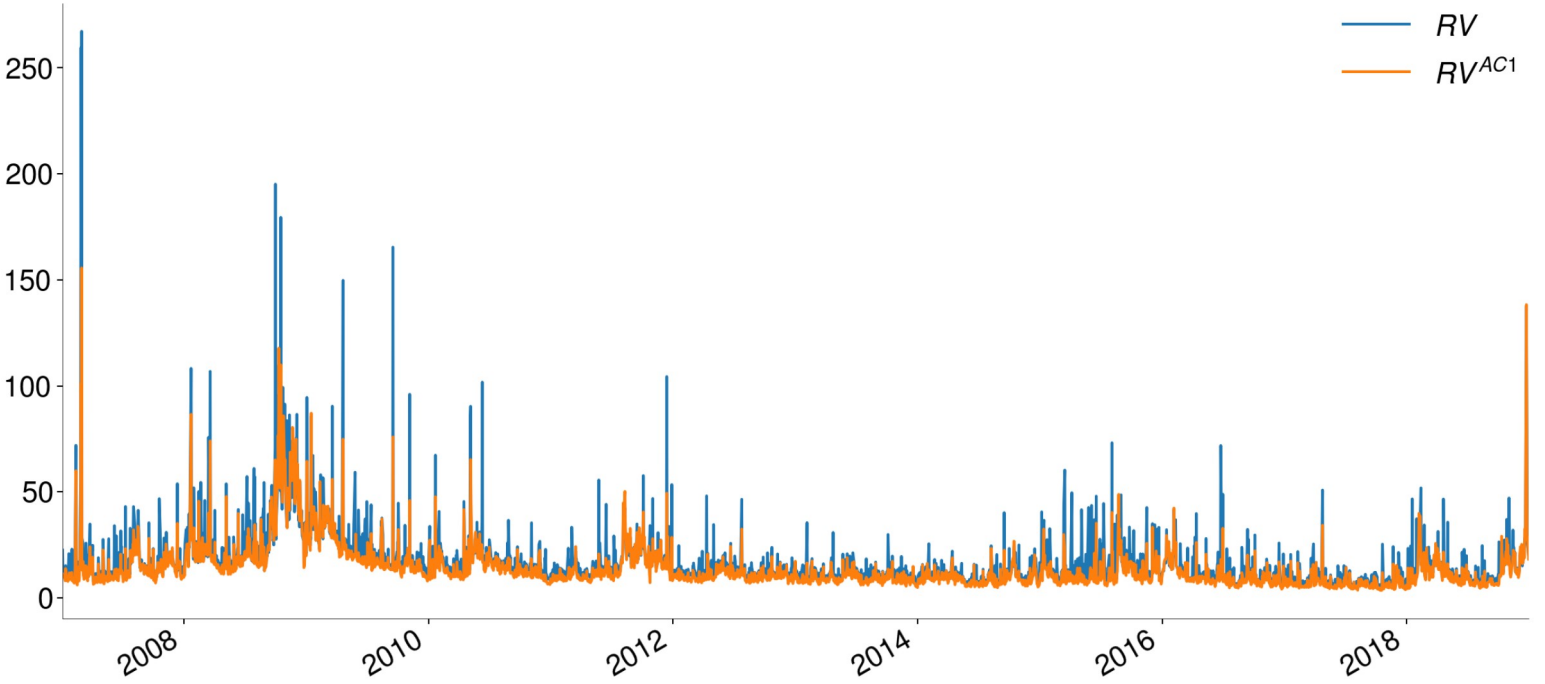


Realized Variance of the S&P 500

- SPY is an ultra-liquid ETF that tracks the S&P 500
- Actual data from 2007 until 2018
- Cleaned and reduced to 1-second data

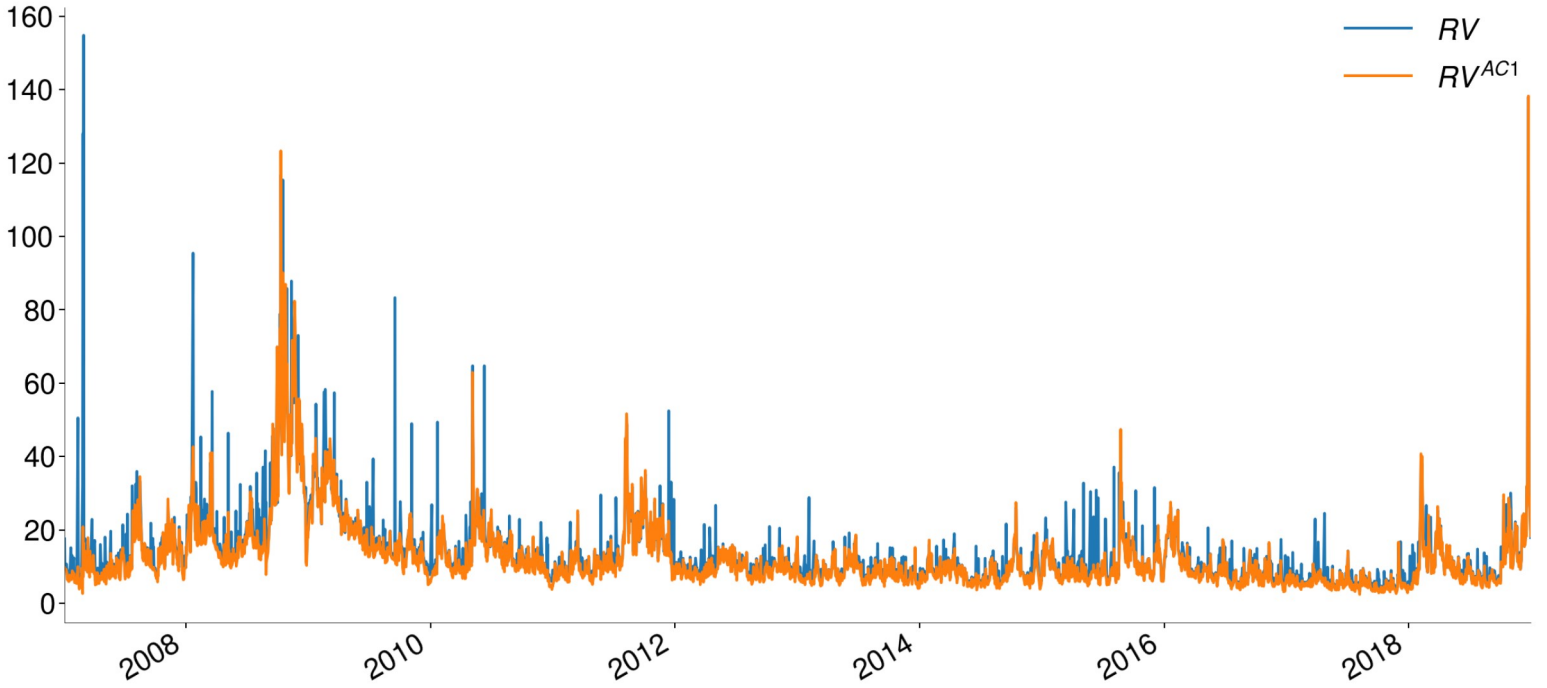
1-second Sampling

In [16]:
```
plot_sp500(0)
```



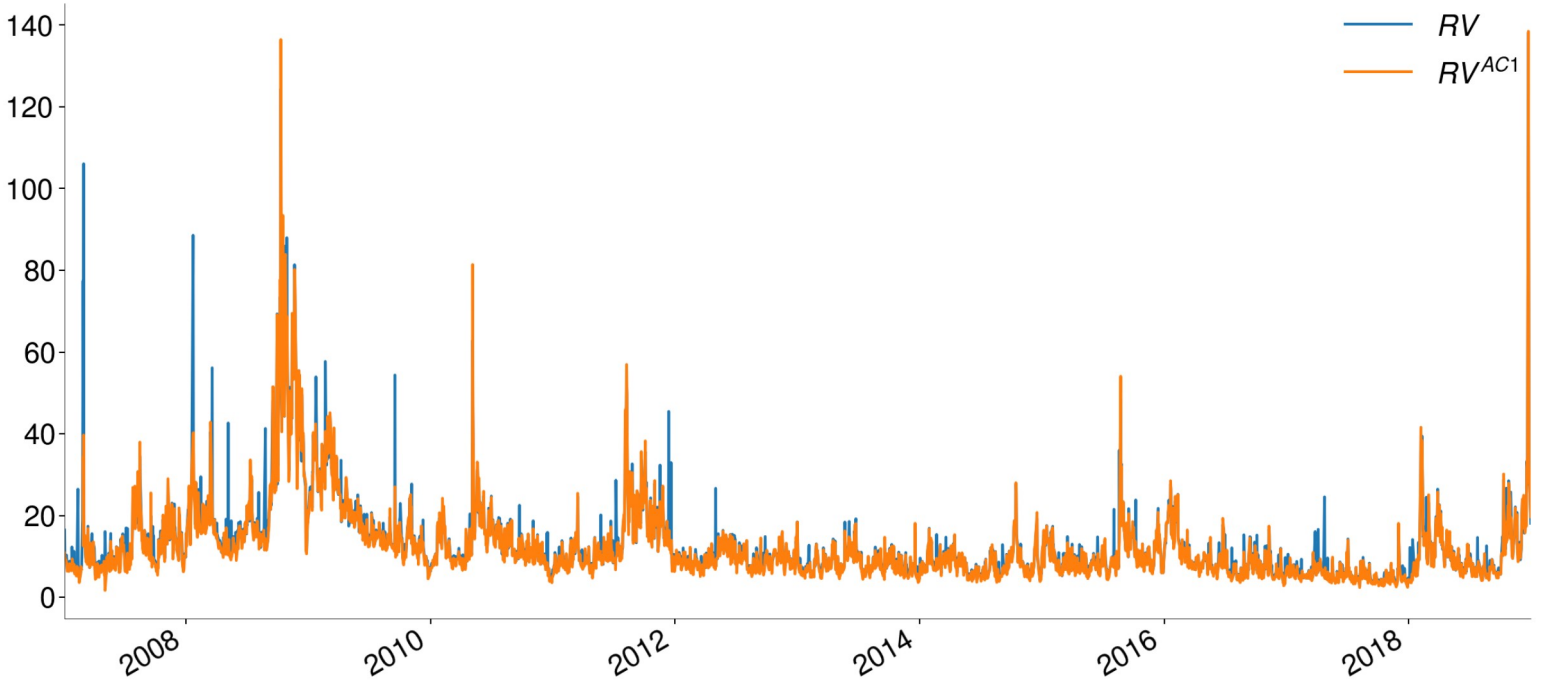
5-second Sampling

In [17]:
```
plot_sp500(1)
```



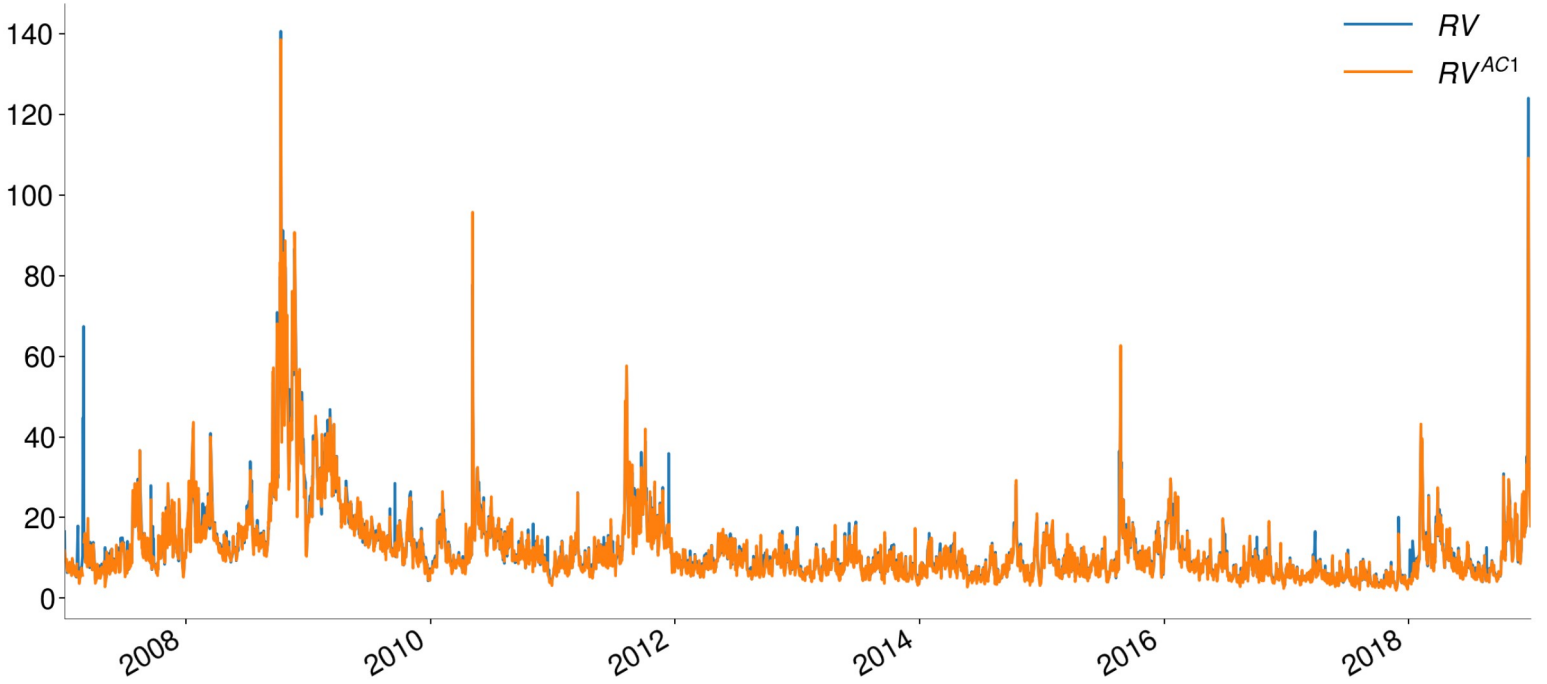
15-second Sampling

In [18]:
```
plot_sp500(2)
```



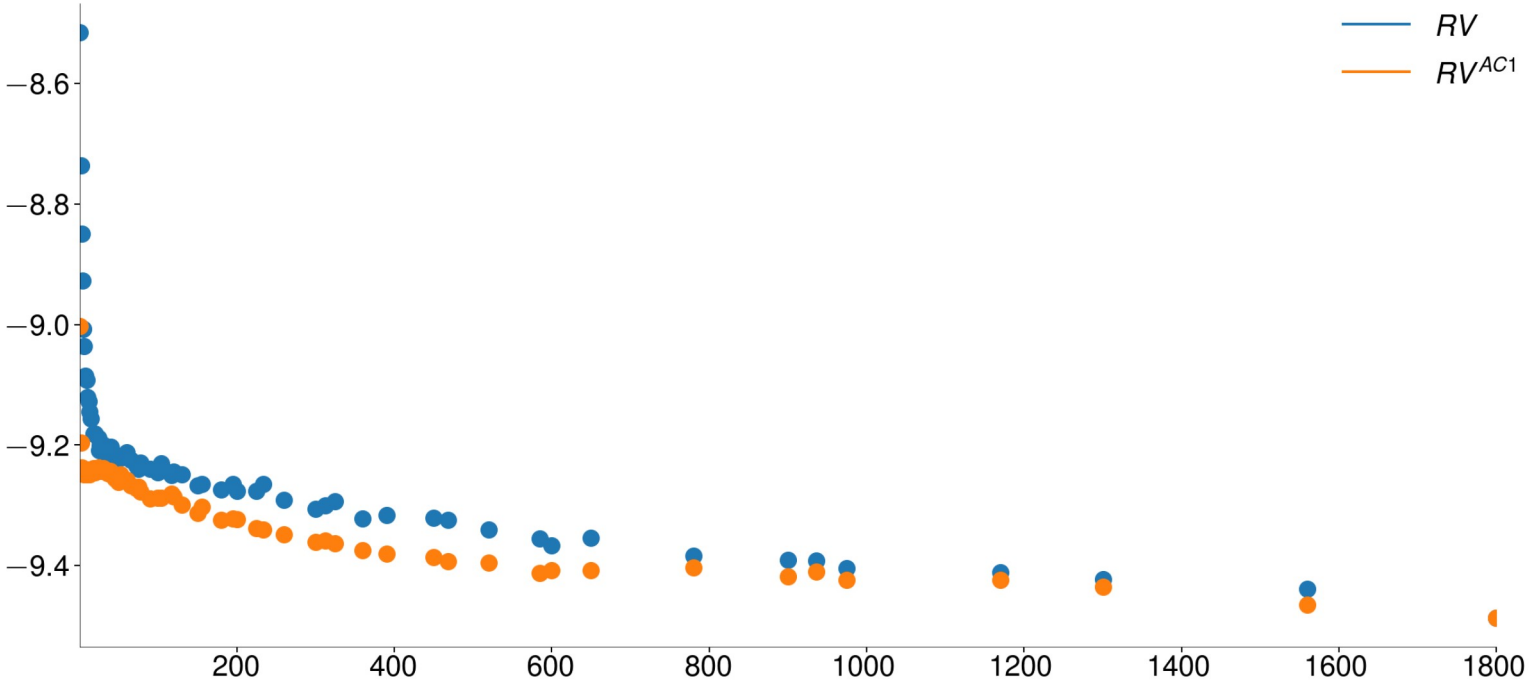
1-minute Sampling

In [19]:
```
plot_sp500(3)
```



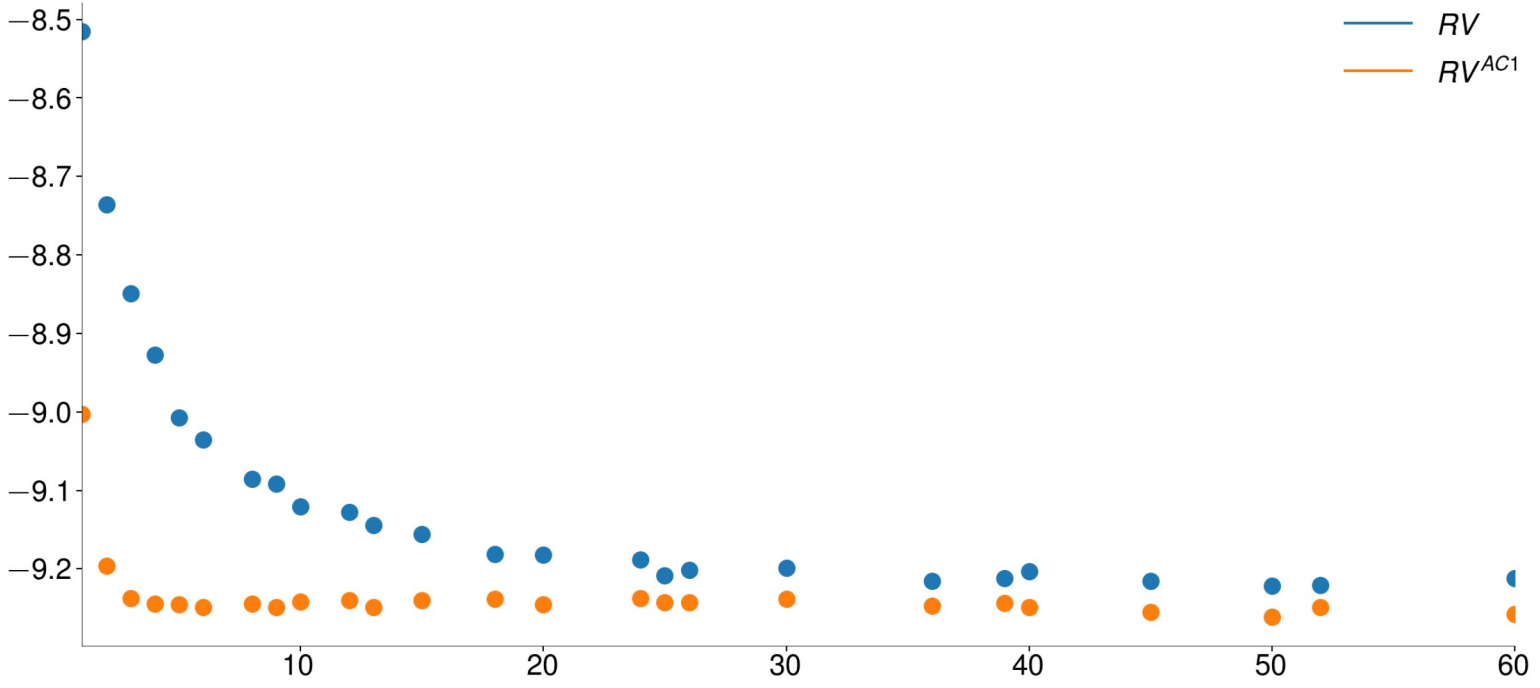
S&P 500 Volatility Signature Plot

In [22]:
```
sp500_plot_vsp()
```



Sub 1-minute Signature

In [23]:
```
sp500_plot_vsp(zoom=True)
```

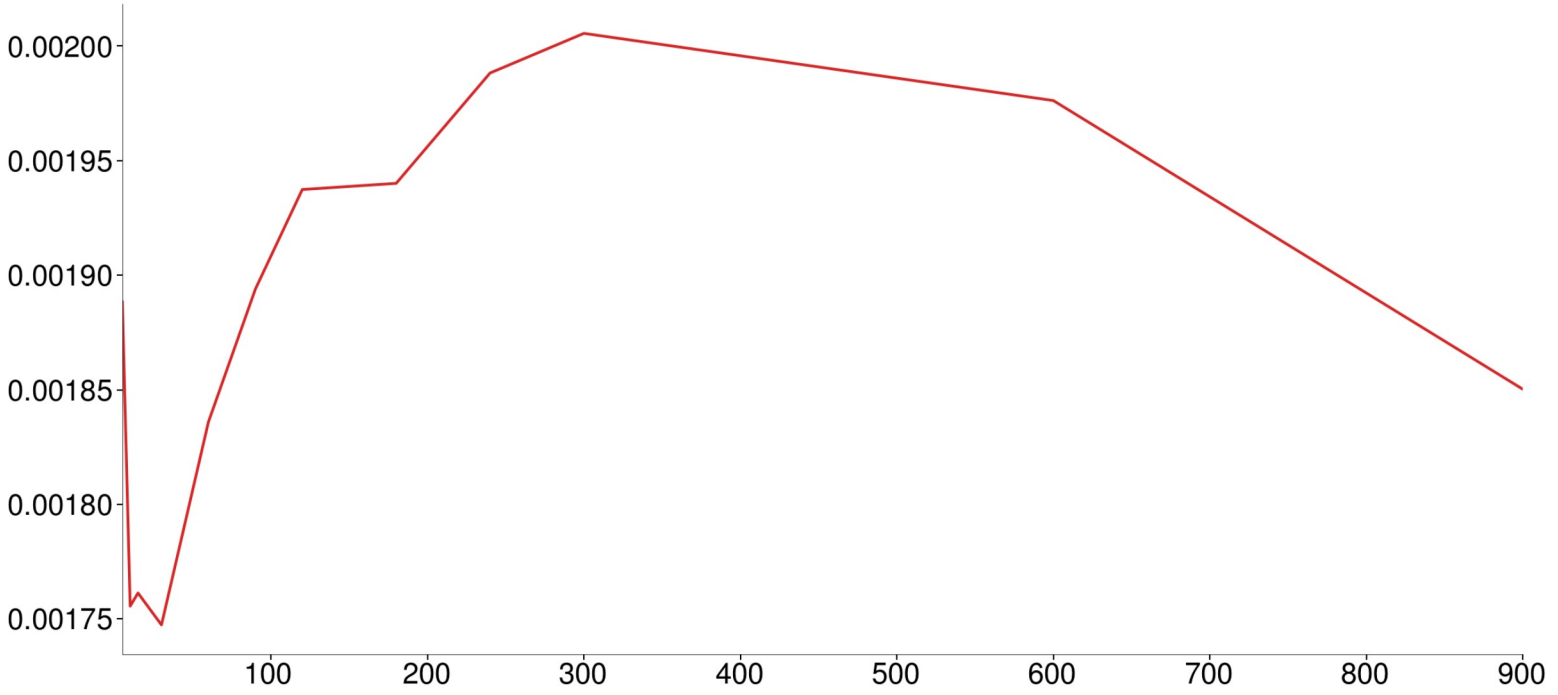


BTC Realized Variance

- BTC is traded in many markets
- Data from Coinbase
- RV works very well in modern assets like BTC

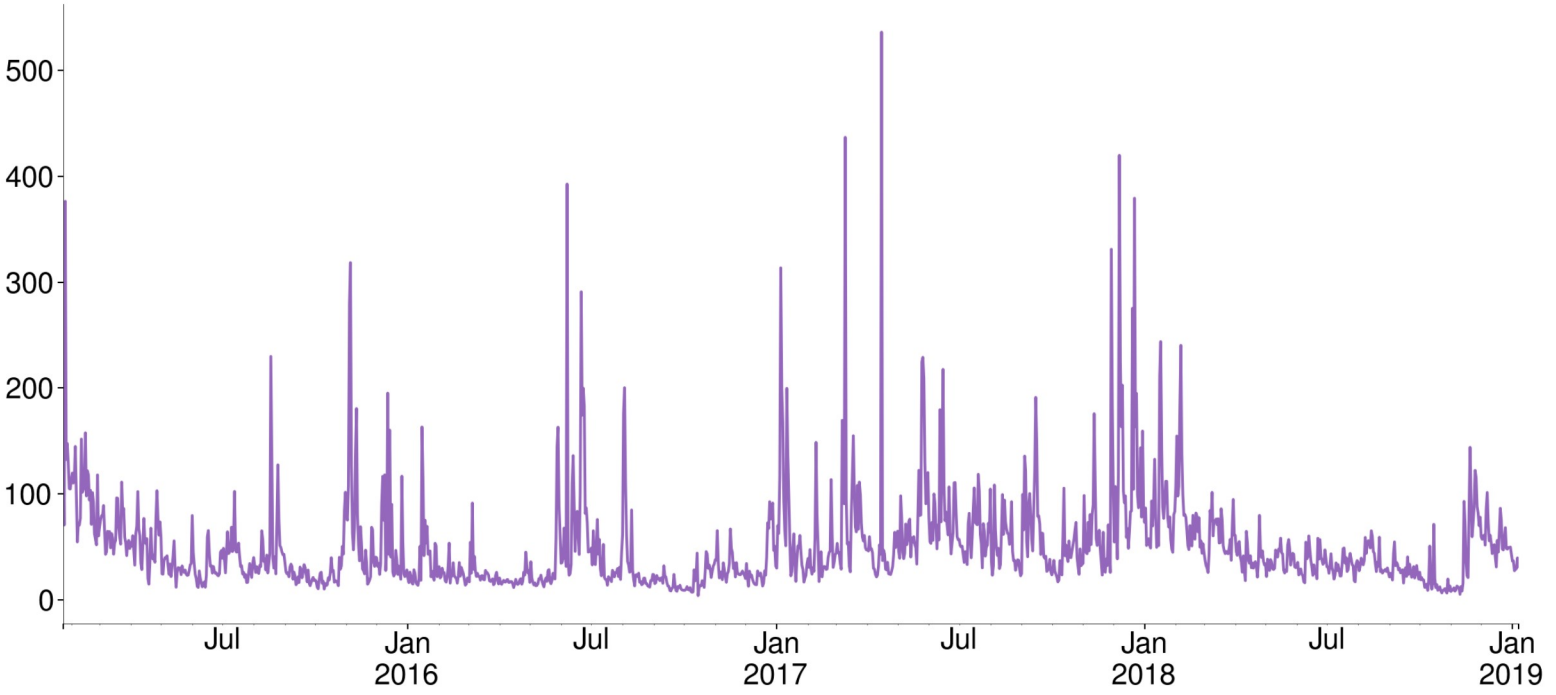
BTC Volatility Signature Plot (Levels)

In [25]:
```
plot_btc_vsp()
```



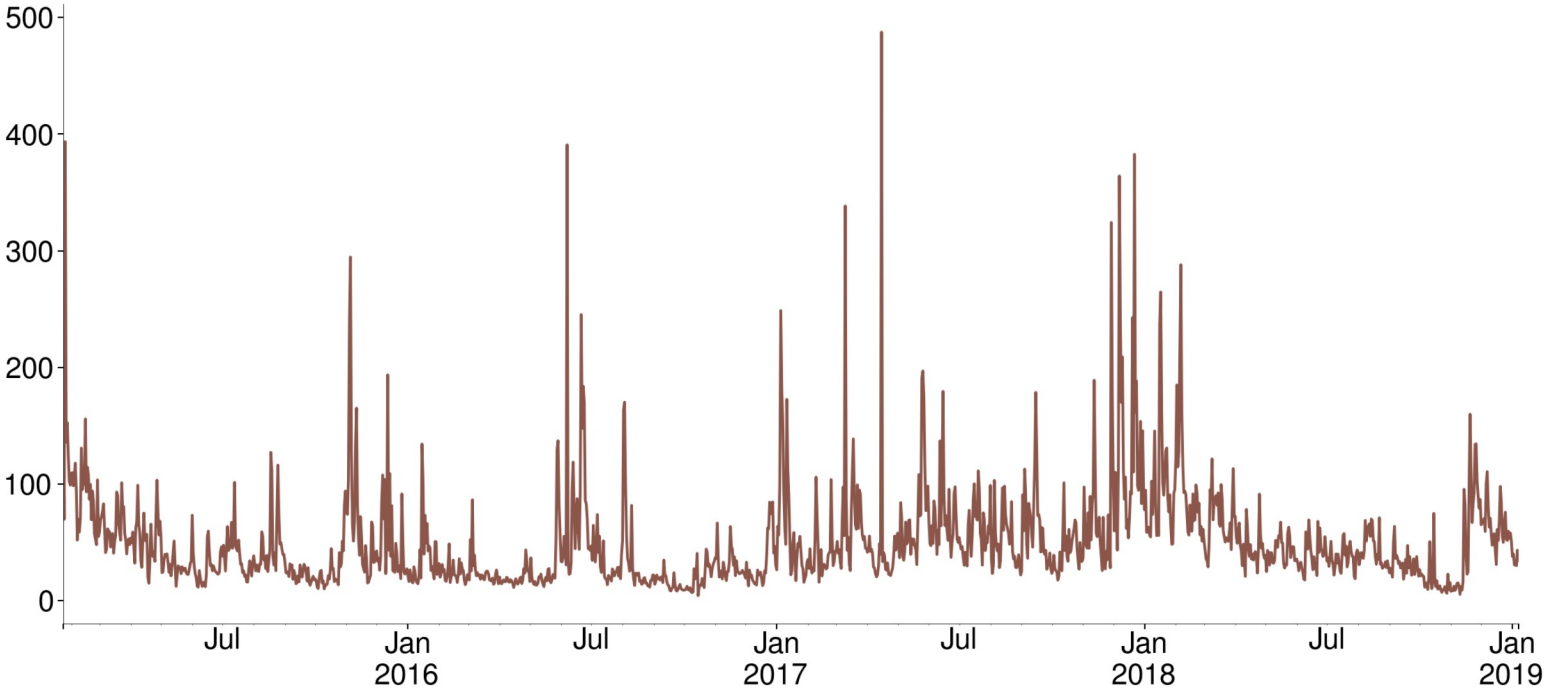
5-second RV

In [27]:
```
plot_btc(0)
```



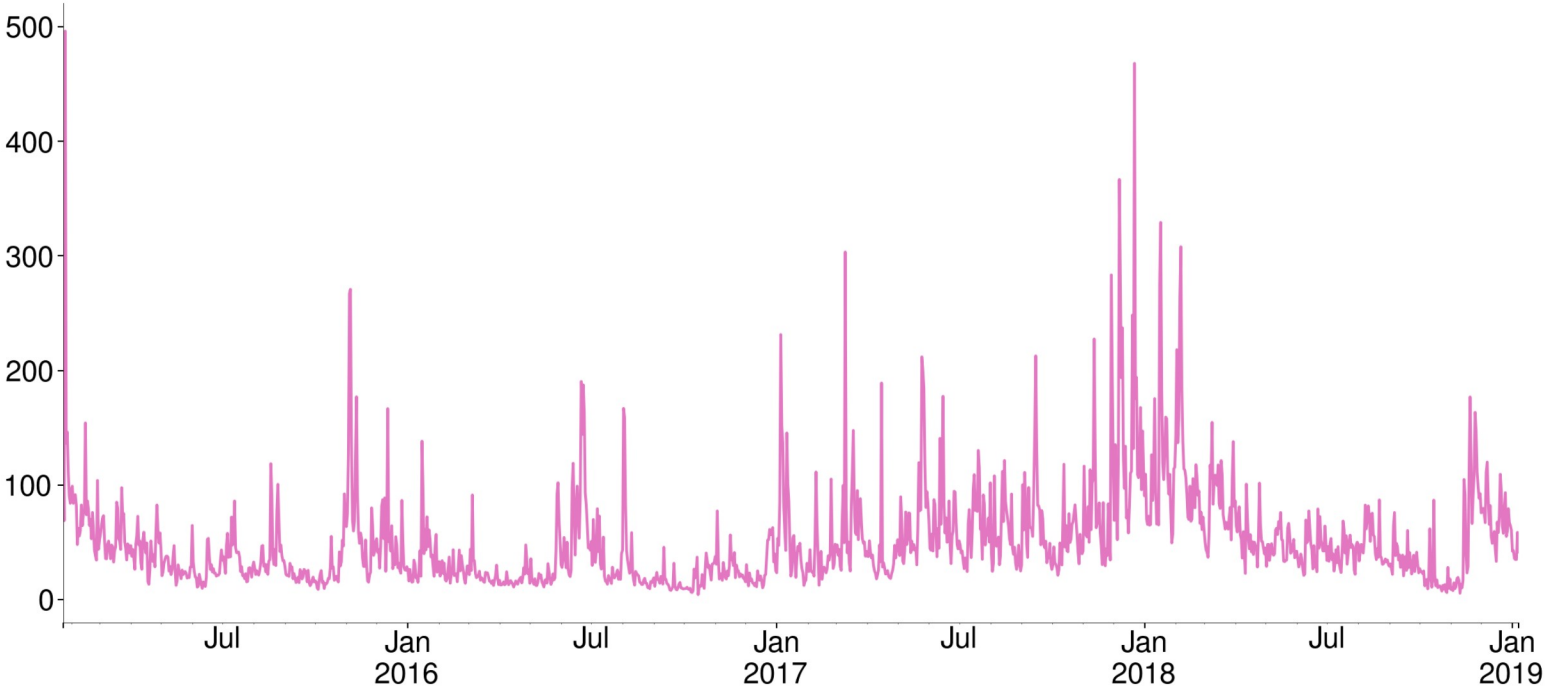
15-second RV

In [28]:
```
plot_btc(1)
```



60-second RV

In [29]:
```
plot_btc(2)
```



Modeling Realized Variance

- Treat as Observable and use ARMA
  - Heterogeneous autoregression (HAR)
- Multiplicative Error Model (MEM)

HAR Models

- Restricted AR(22) in levels

$$RV_t = \phi_0 + \phi_1 RV_{t-1} + \phi_5 \overline{RV}_{5,t-1} + \phi_{22} \overline{RV}_{22,t-1} + \epsilon_t$$

- Or in logs

$$\ln RV_t = \phi_0 + \phi_1 \ln RV_{t-1} + \phi_5 \ln \overline{RV}_{5,t-1} + \phi_{22} \ln \overline{RV}_{22,t-1} + \epsilon_t$$

- Uses a j lag moving average

$$\overline{RV}_{j,t-1} = j^{-1} \sum_{i=1}^{j} RV_{t-i}$$

HAR in levels

In [31]:
```
summary(HARX(sp500_60s, lags=[1,5,22], rescale=False).fit())
```

Mean Model

	coef	std err	t	P>|t|	95.0% Conf. Int.
Const	1.1731e-05	5.453e-06	2.152	3.143e-02	[1.045e-06,2.242e-05]
rv[0:1]	0.1341	0.140	0.955	0.340	[-0.141, 0.409]
rv[0:5]	0.4967	0.173	2.869	4.115e-03	[0.157, 0.836]
rv[0:22]	0.2619	0.138	1.904	5.697e-02	[-7.765e-03, 0.532]

HAR in logs

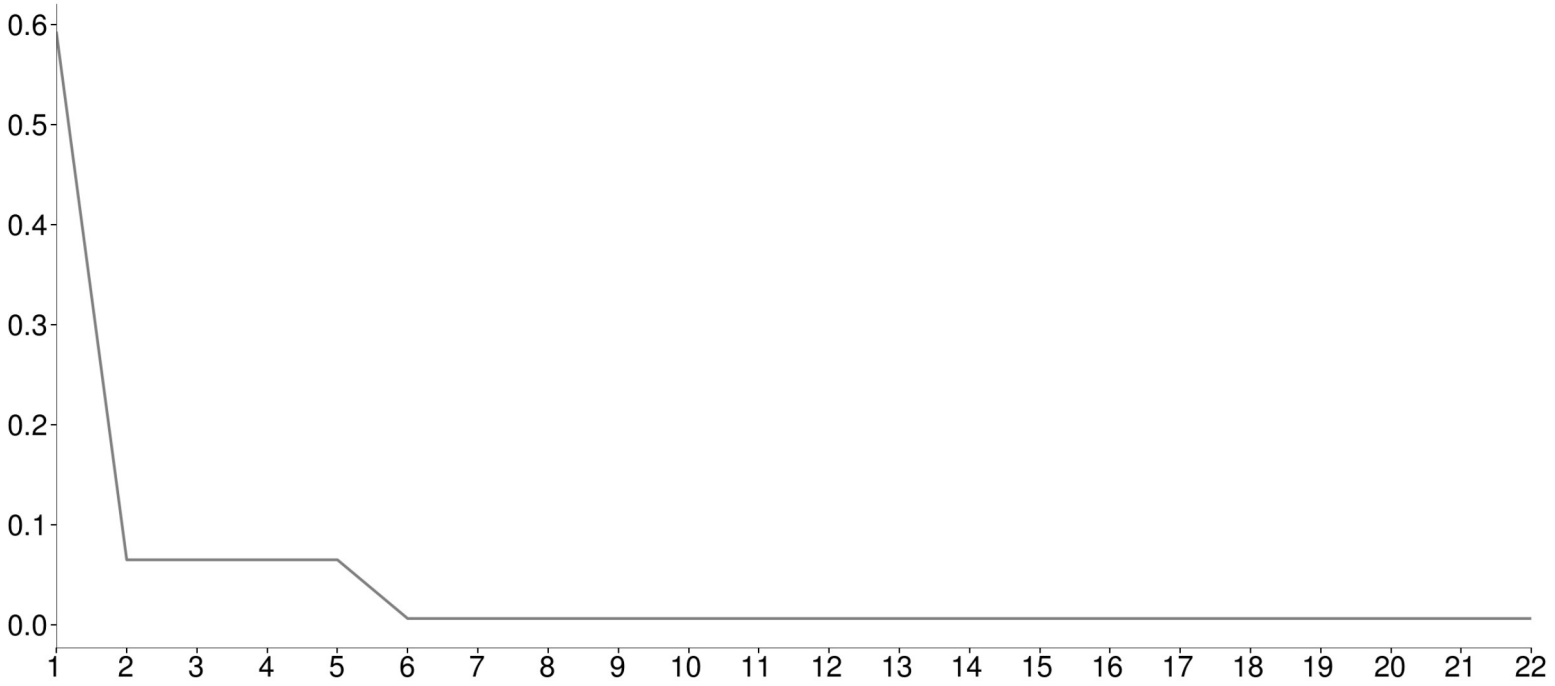
In [33]:
```
summary(HARX(ln_sp500_60s, lags=[1,5,22], rescale=False).fit())
```

Mean Model

	coef	std err	t	P>|t|	95.0% Conf. Int.
Const	-0.4137	9.972e-02	-4.149	3.341e-05	[-0.609, -0.218]
ln RV[0:1]	0.5262	2.536e-02	20.752	1.169e-95	[0.477, 0.576]
ln RV[0:5]	0.2933	3.159e-02	9.284	1.632e-20	[0.231, 0.355]
ln RV[0:22]	0.1390	2.458e-02	5.654	1.566e-08	[9.080e-02, 0.187]

HAR Weights

In [35]:
```
har_weights()
```



Forecasting after $\ln$

- When forecasting after using $\ln$ two options
  - Treat errors are normal and use log-normal to forecast
  - Exponential forecast and use *median* to forecast
- Log-normal uses

$$\mathrm{E}[Y_{t+h|t}] = \exp\left(\mathrm{E}_t[\ln Y_{t+h}] + \frac{\mathrm{V}_t[\ln Y_{t+h}]}{2}\right)$$

Constructing Forecasts

In [36]:

```
fcasts = har.forecast(horizon=10)
df = pd.concat([fcasts.mean.iloc[-1], fcasts.variance.iloc[-1]],1)
df.columns = ["Mean","Var"]
df
```

Out[36]:

	Mean	Var
h.01	-8.557239	0.271689
h.02	-8.539637	0.366659
h.03	-8.553043	0.413345
h.04	-8.592687	0.446340
h.05	-8.628181	0.477080
h.06	-8.626100	0.510716
h.07	-8.630428	0.539273
h.08	-8.639355	0.563083
h.09	-8.651733	0.583799
h.10	-8.661265	0.602613

Finishing Forecasts

In [37]:

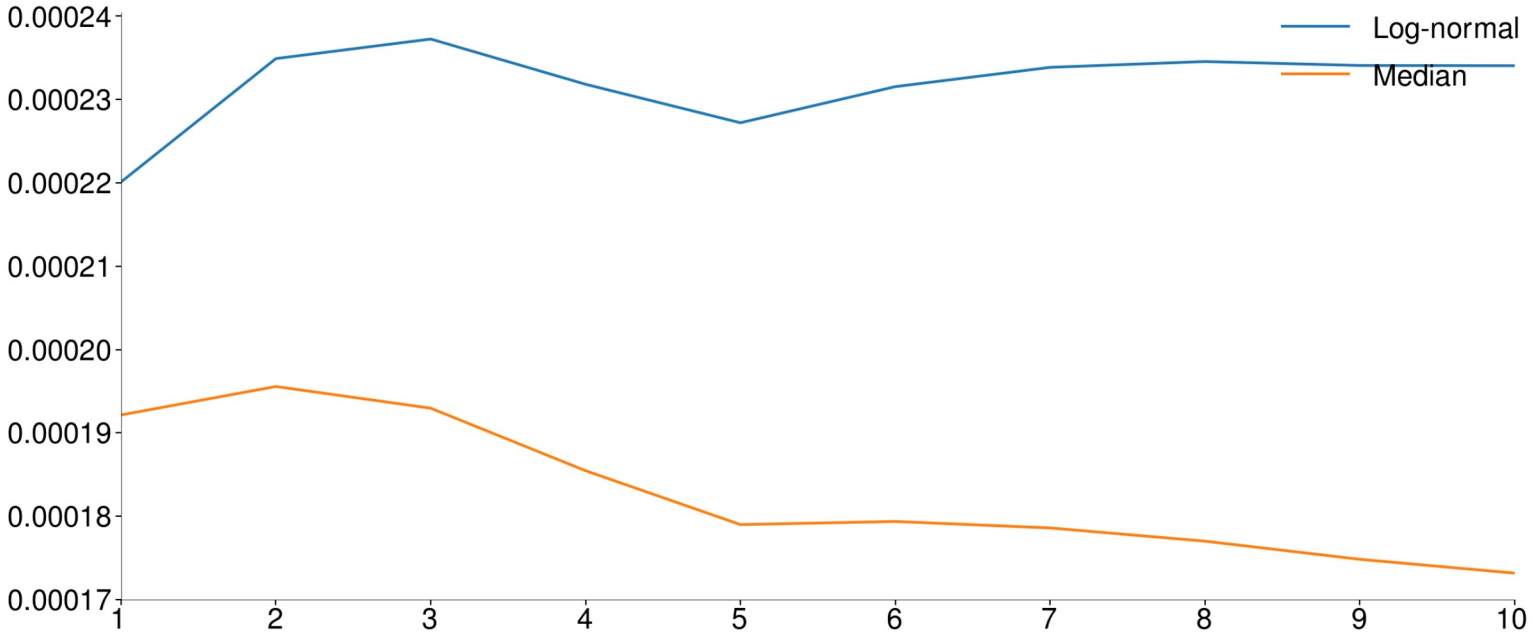
```
both = pd.concat([np.exp(df.Mean + df.Var/2), np.exp(df.Mean)],1)
both.columns = ["Log-normal","Median"]
both
```

Out[37]:

	Log-normal	Median
h.01	0.000220	0.000192
h.02	0.000235	0.000196
h.03	0.000237	0.000193
h.04	0.000232	0.000185
h.05	0.000227	0.000179
h.06	0.000232	0.000179
h.07	0.000234	0.000179
h.08	0.000235	0.000177
h.09	0.000234	0.000175
h.10	0.000234	0.000173

Visualizing forecasts

In [38]:
```
both.index = np.arange(1,11)
plot(both, 14)
```



MEM Models

- MEMs specify the mean of a process as $\mu_t \times \psi_t$ where ψ_t is a mean 1 shock.
- ARCH models are special cases of a non-negative MEM model
- Easy to model RV using existing ARCH mdoes

1. Construct

$$\tilde{r}_t = \text{sign}\left(r_t\right)\sqrt{RV_t}$$

2. Use standard ARCH model building to construct a model for $\tilde{r}_t$

$$\sigma_t^2 = \omega + \alpha_1 \tilde{r}_{t-1}^2 + \gamma_1 \tilde{r}_{t-1}^2 I_{[\tilde{r}_{t-1}<0]} + \beta_1 \sigma_{t-1}^2$$

becomes

$$\sigma_t^2 = \omega + \alpha_1 RV_{t-1} + \gamma_1 RV_{t-1} I_{[r_{t-1}<0]} + \beta_1 \sigma_{t-1}^2$$

GARCH MEM

In [40]:

```
from arch import arch_model
res = arch_model(np.sqrt(sp500_60s),mean="zero",o=0, rescale=True).fit(disp="off")
summary(res)
```

Out[40]:

Zero Mean - GARCH Model Results

Mean Model:	Zero Mean	**Log-Likelihood:**	-3267.13
Vol Model:	GARCH	**AIC:**	6540.26
Distribution:	Normal	**BIC:**	6558.30

Volatility Model

	coef	std err	t	P>|t|	95.0% Conf. Int.
omega	0.0261	4.965e-03	5.255	1.484e-07	[1.636e-02,3.582e-02]
alpha[1]	0.7256	5.885e-02	12.331	6.181e-35	[0.610, 0.841]
beta[1]	0.2744	5.061e-02	5.421	5.925e-08	[0.175, 0.374]

HARCH MEM

In [41]:

```
from arch.univariate import HARCH
mod = arch_model(np.sqrt(sp500_60s),mean="zero",o=0, rescale=True)
mod.volatility = HARCH([1,5,22])
res = mod.fit(disp="off")
summary(res)
```

Out[41]:

Zero Mean - HARCH Model Results

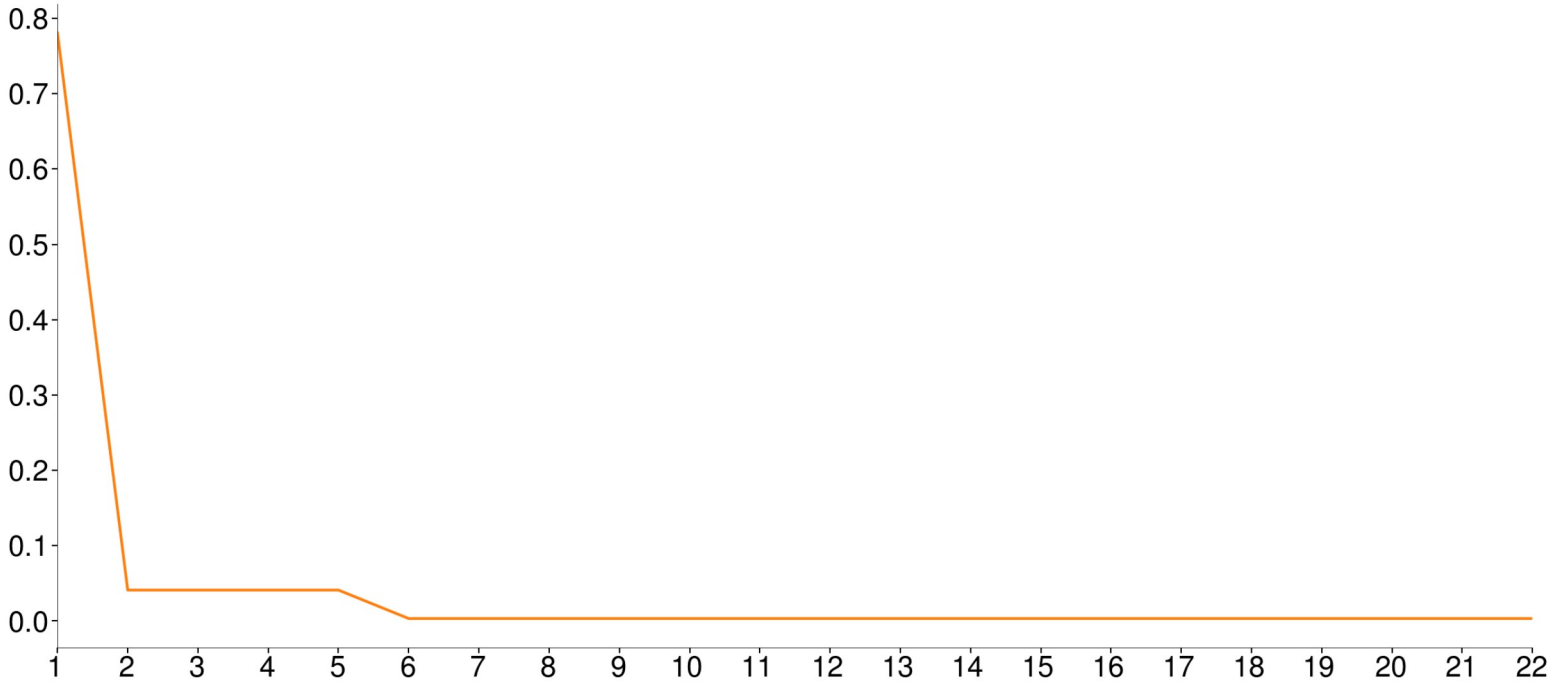
Mean Model:	Zero Mean	**Log-Likelihood:**	-3267.30
Vol Model:	HARCH	**AIC:**	6542.60
Distribution:	Normal	**BIC:**	6566.65

Volatility Model

	coef	std err	t	P>|t|	95.0% Conf. Int.
omega	0.0305	6.020e-03	5.073	3.909e-07	[1.874e-02,4.234e-02]
alpha[1]	0.7395	5.959e-02	12.410	2.298e-35	[0.623, 0.856]
alpha[5]	0.1888	4.660e-02	4.052	5.080e-05	[9.749e-02, 0.280]
alpha[22]	0.0717	2.653e-02	2.702	6.887e-03	[1.969e-02, 0.124]

HARCH Weights

In [43]:
```
harch_weights()
```



Implied Variance

- Implied volatility is very different from ARCH and Realized measures
- Market based: Level of volatility is calculated from options prices
- Forward looking: Options depend on future price path
- “Classic” implied relies on the Black-Scholes pricing formula
- “Model free” implied volatility exploits a relationship between the second derivative of the call price with respect to the strike and the risk neutral measure
- VIX is a Chicago Board Options Exchange (CBOE) index based on a model free measure
- Allows volatility to be directly traded

Black-Scholes Implied Volatility

- Prices follow a geometric Brownian Motion

$$\mathrm{d}S_t = \mu S_t \mathrm{d}t + \sigma S_t \mathrm{d}W_t$$

- Constant drift and volatility
- Price of a call is

$$C(T,K) = S\Phi(d_1) + Ke^{-rT}\Phi(d_2)$$

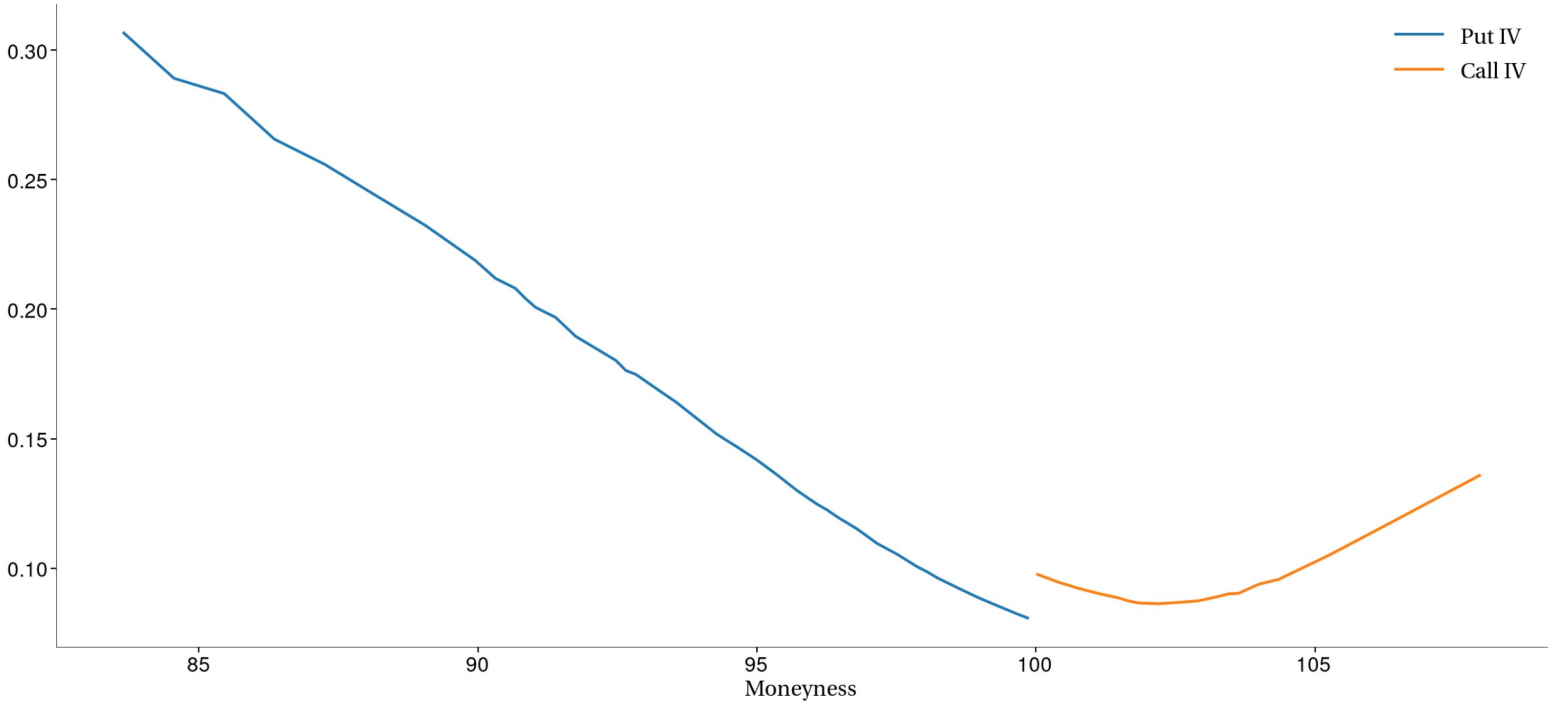
$$d_1 = \frac{\ln(S/K) + \left(r + \sigma^2/2\right)T}{\sigma\sqrt{T}},\ d_2 = \frac{\ln(S/K) + \left(r - \sigma^2/2\right)T}{\sigma\sqrt{T}}$$

- Invert to produce a formula for the volatility given the call price $C(T,K)$

$$\sigma_t^{Implied} = g\left(C_t(T,K), S_t, K, T, r\right)$$

Black-Scholes Smile

In [45]:
```
bsiv()
```



Model-free Implied Variance

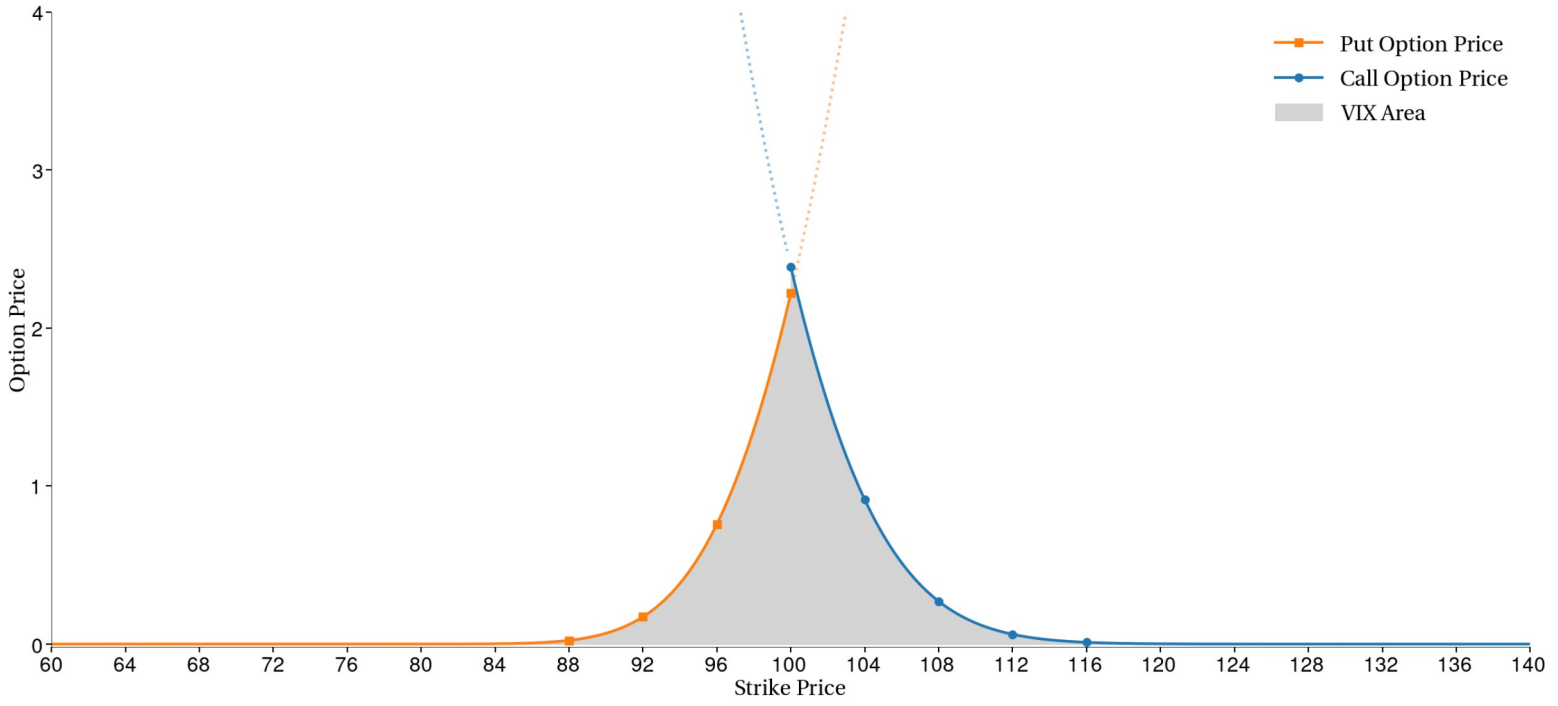
- VIX is continuously computed by the CBOE
- Uses both out-of-the-monry calls and puts

$$\sigma^2 = \frac{2}{T} e^{rT} \sum_{i=1}^{N} \frac{\Delta K_i}{K_i} \frac{Q(K_i)}{K_i} - \frac{1}{T} \left(\frac{F_0}{K_0} - 1 \right)^2$$

- $Q(K_i)$ is the mid-quote for a strike of K_i, K_0 is the first strike below the forward index level
- VIX appears to have information about future realized volatility that is not in other backward looking measures (GARCH/RV)
- Computes area under curves defined by OOM options

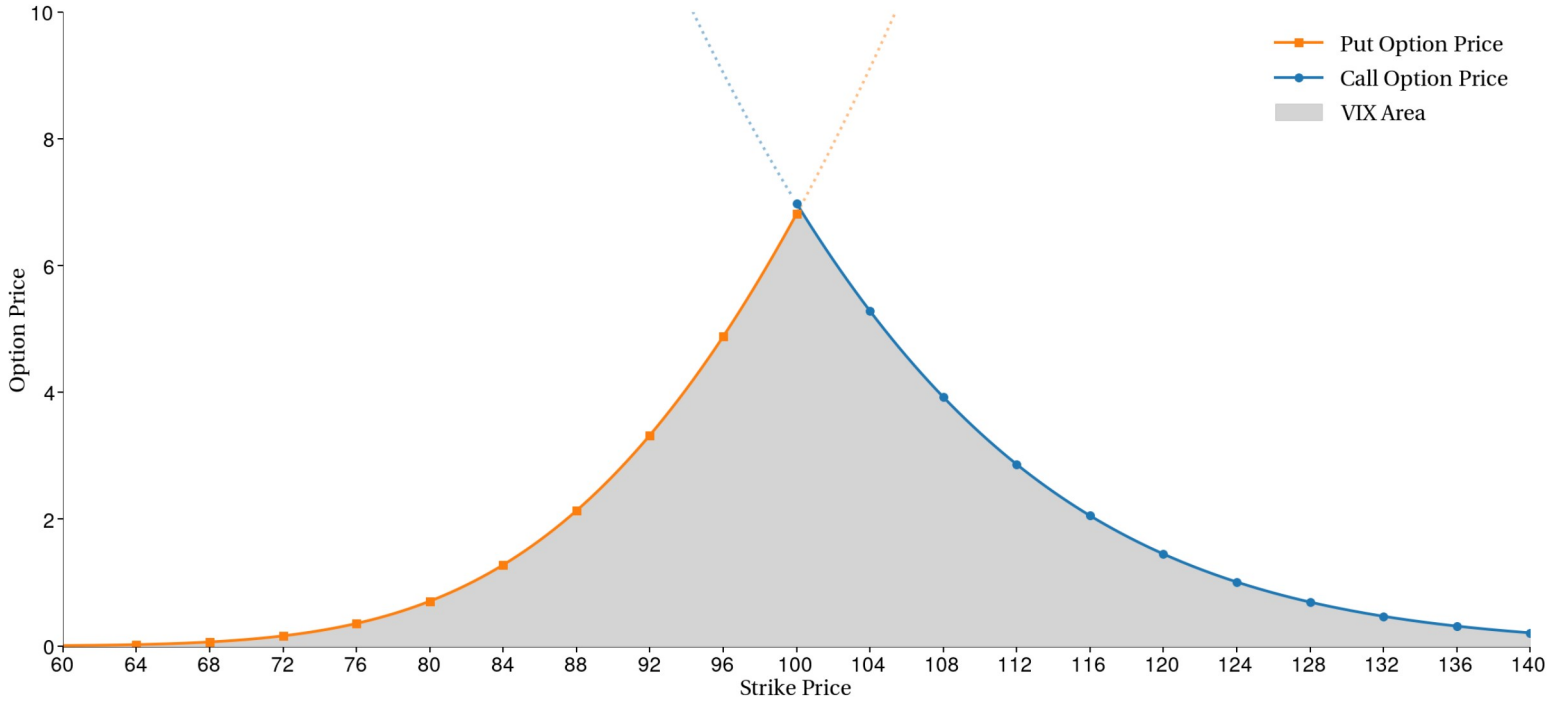
20% Annualized Volatility

In [47]:
```
plot_20()
```



60% Annualized Volatility

In [49]:
```
plot_60()
```



Next Week

- Risk Management
  - Value-at-Risk
  - Expected Shortfall